Low-Latency Plesiochronous Data Retiming [1]

Larry R. Dennison, William J. Dally, Duke Xanthopoulos
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

A new method of retiming plesiochronous data is described. This method features latency of less than a cell-time and requires only minimal support circuitry. No flow control or handshaking signals are used, allowing true undirectional signalling between transmitter and receiver.

Application areas include communication networks in parallel computers, and general communication network repeaters, hubs, bridges, and routers.

1: Introduction

The Reliable Router [1] project at MIT is developing a VLSI communications chip for use in large parallel computers. The research goals of the project have been on exploring mechanisms to aid in the design of large, high-performance, reliable systems.

One of the many problems facing the designer of a large digital system is clock distribution. The large system is typically composed of communicating subsystems. These subsystems are internally synchronous and clocking uncertainty within a subsystem is well-controlled. Between subsystems, communications becomes more difficult, owing to larger amounts of clock uncertainty. As clock speeds increase, the uncertainty becomes the performance limiting factor.

In addition, the large clock tree is a single point of failure. The builder of a large system tries to avoid single points of failure wherever possible. The designer may not be trying to achieve complete fault-tolerance, but the goals of graceful degradation and hot-plug-in are very important and conflict with a single clock distribution system.

Existing solutions to this problem include asynchronous and mesochronous (same clock source, unknown clock phase) timing methods. Messerschmitt [2] offers a good discussion of these terms. None of these methods seemed particularly well-suited for use in the router. Asynchronous timing requires a synchronizer delay for each data item. Mesochronous timing involved some small amount of delay but still required a single system clock source.

A third method, plesiochronous timing, looked promising, as demonstrated in well-known embodiments such as SONET. It removed the need for a single clock and requires

[1] This research is supported by the Advanced Research Projects Agency under contracts N00014-88K-0738, F19628-92-C-0045, and N00014-91-J-1698.

no flow control handshaking. The difficulty with plesiochronous timing is keeping the data items transferred one-for-one between transmitter and receiver. Dependent on the relative rates, transmitted data may be either undersampled or oversampled by the receiver. Previous designs used fairly deep FIFOs ($>$ 4 elements!) and had complex finite state machines. In general, prior designs use FIFOs of a depth large enough to compensate for a synchronizer delay.

The solution described in this paper separates the synchronizer from the data retiming, giving minimum data latency and minimum data storage requirements. By carefully chosing the point at which a data retiming adjustment takes place, data items are kept one-for-one without requiring any additional circuitry.

This solution does differ from other synchronizer-avoidance methods, as it addresses the issue of keeping the data items one-for-one between transmit and receive clock domains. Glasser and Rettberg [3] use dynamic delay adjustment to avoid the synchronizer penalty on the data path, but rely on mesochronous timing to keep the data one-for-one. Stewart and Ward [4] extend the idea of synchronizer avoidance to cover plesiochronous timing, but do not address the issue of keeping data one-for-one while providing minimal latency.

2: Plesiochronous Requirements

A pleisochronous system is one where all the clocks operate at approximately the same frequency, f_0. In point-to-point communications, there is a transmitter operating at frequency f_t and a receiver operating at frequency f_r. Both f_t and f_r are in the interval $[f_0 - \Delta f, f_0 + \Delta f]$.

If the transmitter were to send data to the receiver at the transmitter's clock rate, one of two things will eventually happen, dependent on the relative speeds of the two clocks. If the transmitter is running faster than the receiver ($f_t > f_r$), the receiver will be presented with more information than it can handle causing an overrun condition. If the receiver is running faster ($f_t < f_r$), it becomes starved resulting in an underrun condition.

To avoid underruns and overruns, the transmitter and receiver agree that the transmitter will not always send data. The transmitter simply sends data at rate lower than the receiver is operating. The receiver distinguishes between data and non-data, and only processes data. More formally, the transmitter produces a constant stream of cells, where each cell can contain either data or non-data. The rate at which the stream of cells is produced is defined to be the transmitter's frequency. The rate at which the receiver is able to consume data cells is defined to be the receiver's frequency. As long as the transmitter does not send data cells at rate faster than the receiver's consumption rate, no overrun condition will exist. If the receiver can tolerate an occasional delay in the arrival of the next data cell, no underrun condition will exist.

In a system where communication occurs by linking up several point-to-point hops, all transmitters in the system must produce data at a rate lower than the slowest receiver frequency. This is not as bad as it sounds, as this is usually nothing more than the worst-case transmitter/receiver frequency mismatch. Since the transmitter does not know the relative frequencies, it presumes that its clock is fast and the receiver is slow ($f_t = f_0 + \Delta f, f_r = f_0 - \Delta f$). To avoid the overrun, the transmitter sends data at the rate

$$f_d = f_t \frac{f_0 - \Delta f}{f_0 + \Delta f} \tag{1}$$

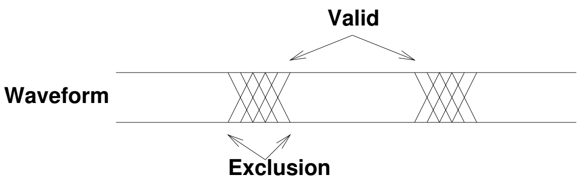

Figure 1: Valid and Exclusion Regions

When $f_t > f_r$, the transmitted data rate exactly matches the receiver's consumption rate. When $f_t < f_r$, the data rate is not matched and a degradation occurs which is approximately $2\Delta f$.

3: Retiming

Retiming is the movement of a cell from the transmitter's clock domain to the receiver's clock domain. The incoming cells have a portion of the cell time (t_{clk}) where their value is stable and can be sampled by a flip-flop using some clock edge. This is called the valid region, t_{val}. Over the rest of the cell, either the value of the cell is changing or the value has not met the setup or hold times of a flip-flip. This region is called the exclusion region, t_{exc}. By definition,

$$t_{clk} = t_{val} + t_{exc}$$

For ease of explanation, the width of the valid region is assumed to be much larger than the width of the exclusion region ($t_{val} \gg t_{exc}$). A waveform with both regions illustrated is shown in figure 1

In a synchronous system, the cell can be safely retimed into the receiver's clock domain if the receiver's clock edge occurs during the valid region. The design of a correct synchronous system is guaranteeing that the clock edge occurs exactly in that region.

3.1: Mesochronous Retiming

For a mesochronous system, the placement of the receiver's clock edge relative to the valid region is not controlled. In fact, the clock edge could occur during the exclusion region. To get around this, a waveform is constructed which is simply the transmitted waveform delayed by half a clock period. For notational ease, the original transmitted waveform will be called the Q-waveform, the delayed will called the R-waveform[2].

It easy to see that if $t_{val} > t_{clk}/2$, at any point in time either the original waveform or the delayed waveform are in the valid region. Now, if the receiver clock edge occurs in the exclusion region of the Q waveform, the R waveform is sampled instead. To build a complete mesochronous retiming circuit, one needs to construct a subcircuit which delays

[2] Q because it comes from the Q output of a flip-flop in the implementation, R because it came after Q

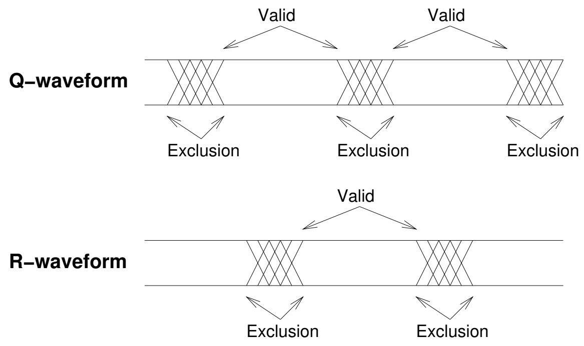

Figure 2: Q and R waveforms

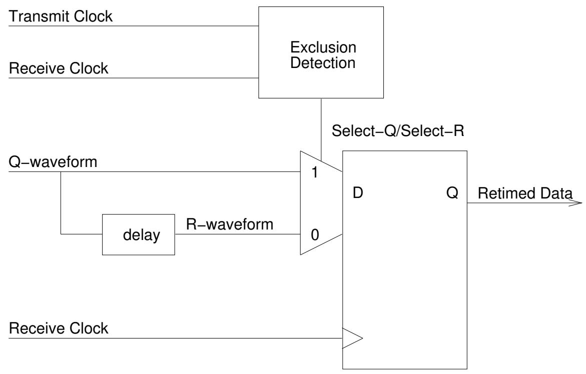

Figure 3: Mesochronous Retiming Circuit

the transmitted cell, a circuit for detecting sampling during the exclusion region, and a multiplexor. A block diagram is shown in figure 3.

When $t_{val} < t_{clk}/2$, multiple delayed versions of the incoming waveform are required. In general, one needs $\lceil \frac{t_{clk}}{t_{val}} \rceil$ versions.

3.2: Plesiochronous Retiming

Plesiochronous retiming looks very similar to mesochronous retiming, except the relative phase Θ between the transmit and receive clocks will vary with time

$$\Theta(t) = \Theta_0 + 2\pi(f_t - f_r)t \tag{2}$$

To extend the mesochronous case to the plesiochronous case, one could simply allow dynamic switching between the Q and R waveforms. However, indiscriminate switching can result in either duplicate cells or missing cells. Figure 4 shows a case where the receiver has been sampling the Q waveform. The receive clock is running slower than the transmit clock, so the receive clock edge encounters Q's exclusion region on the right hand side.

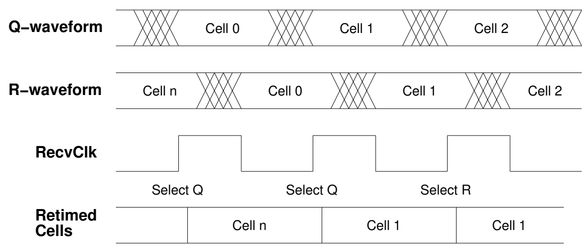

Figure 4: Cell Oversampling

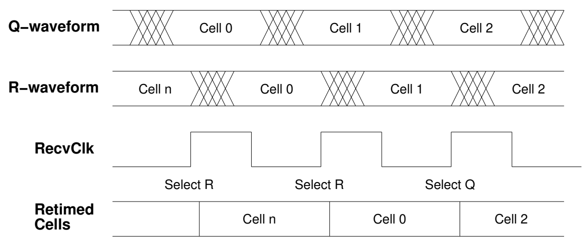

Figure 5: Cell Undersampling

The receiver then shifts to sampling the R waveform, resulting in cell 1 appearing twice at the ouput of the receiver.

Similarly, figure 5 shows a case where the receiver has been sampling the R waveform. The receive clock is running fast relative to the transmit clock, so the receive clock bumps into R's exclusion region on the left hand side. The receiver then shifts to sampling the Q waveform, causing the receiver output to skip over cell 1.

3.3: Correct Plesiochronous Retiming

Suppose one wanted to control the switching between the Q and R waveforms using an automaton clocked in the transmit domain. The phase of the transmit clock relative to the receive clock is unknown, so the best time to flip the control input to the multiplexor is when both Q and R are valid and have the same value. This occurs at $\frac{3\pi}{2}$ of the way into a cell, measured on the Q waveform. Figure 6 shows the correct point. Of course, the multiplexor should be hazard free. When the circuits are done properly, the receiver is always sampling a waveform during its valid region.

To solve undersampling/oversampling, recall that cells are either data or non-data. It is perfectly acceptable to insert or delete non-data cells, but it would be bad form to insert or delete data cells. If one restricts switching between the Q and R waveforms to the times when both Q and R contain a non-data cell, only non-data cells will be undersampled or oversampled.

There are four cases to consider:

- The receive clock is faster than the transmit clock, and the automaton switches from

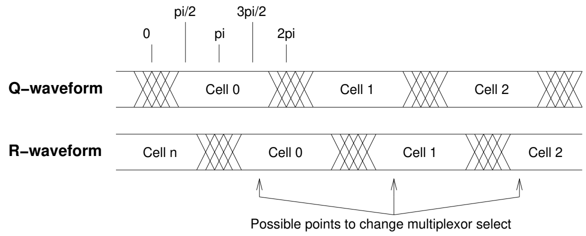

Figure 6: Multiplexor Control Switching Time

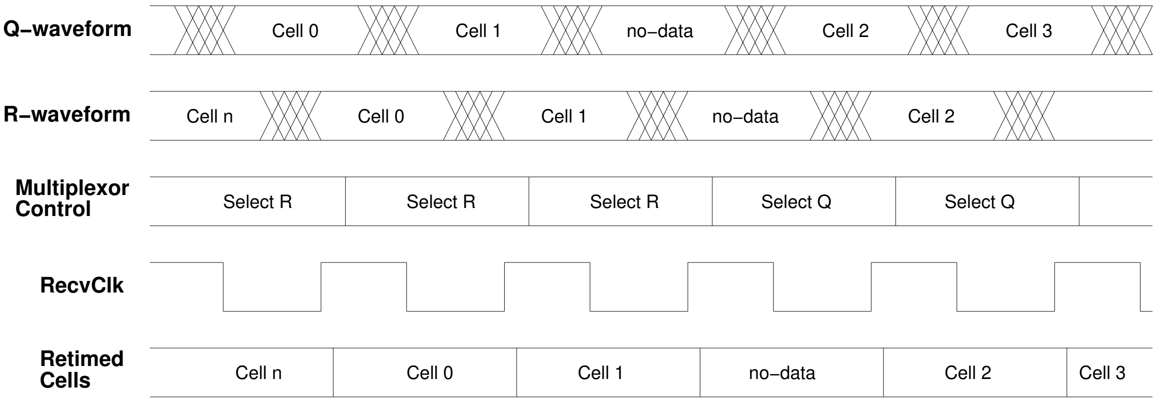

Figure 7: RxClk fast, $R \rightarrow Q$

R to Q.

- The receive clock is faster than the transmit clock, and the automaton switches from Q to R.
- The receive clock is slower than the transmit clock, and the automaton switches from R to Q.
- The receive clock is slower than the transmit clock, and the automaton switches from Q to R.

The case of $f_r > f_t$, $R \rightarrow Q$ is shown in figure 7. The fast receive clock implies that the sampling edge will encounter R's exclusion region on the right hand side. The figure shows that as the switchover is made, no cells are added or dropped. Figure 8 illustrates $f_r > f_t$, $Q \rightarrow R$. The fast clock implies that the sampling edge will encounter Q's exclusion region on the right hand side. The figure shows that as the switchover is made, the non-data cell is duplicated. In combination, these two cases show that when the receive clock is faster than the transmit clock, extra non-data cells are generated by the receiver to compensate.

The case of $f_r < f_t$, $R \rightarrow Q$ is shown in figure 9. The fast receive clock implies that the sampling edge will encounter R's exclusion region on the left hand side. The figure shows that as the switchover is made, the non-data cell is dropped. Figure 10 illustrates $f_r < f_t$, $Q \rightarrow R$. The fast clock implies that the sampling edge will encounter Q's exclusion region

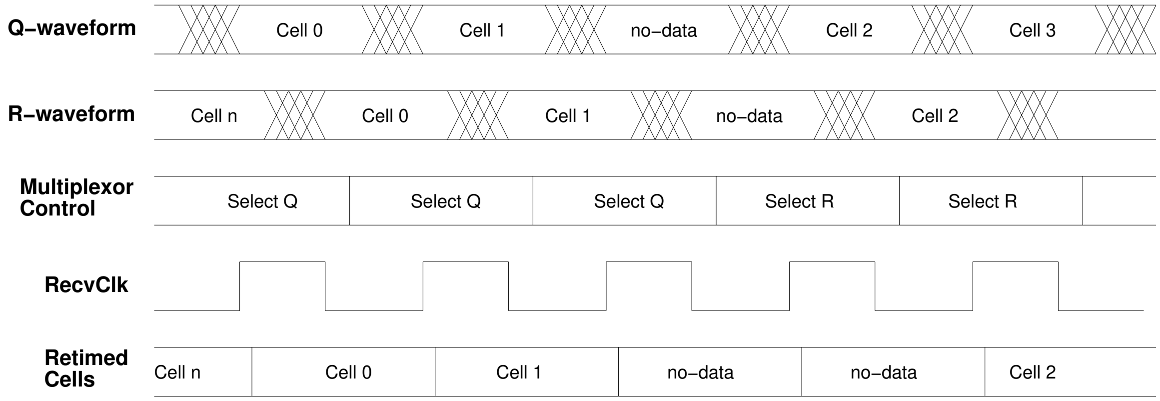

Figure 8: RxClk fast, $Q \rightarrow R$

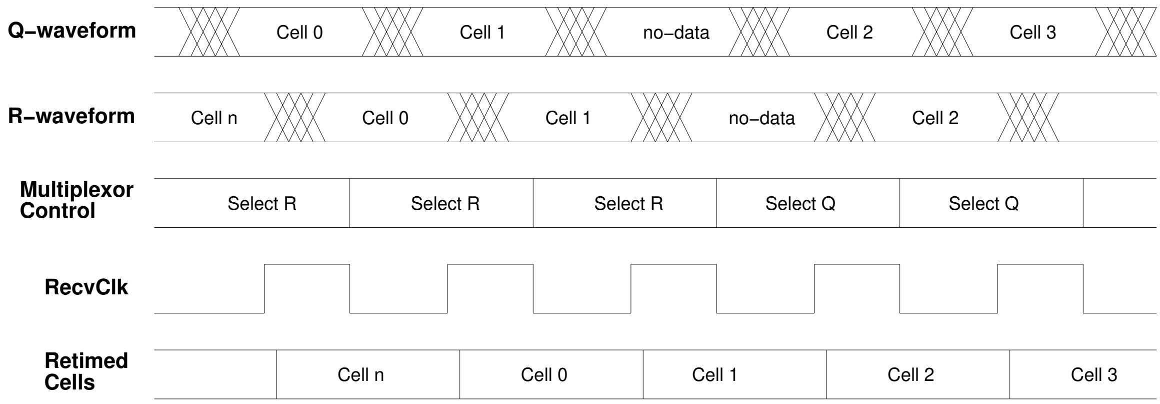

Figure 9: RxClk slow, $R \rightarrow Q$

on the right hand side. The figure shows that as the switchover is made, no cells are added or dropped. In combination, these two cases show that when the receive clock is slower than the transmit clock, non-data cells are deleted by the receiver to compensate.

Knowing when Q and R contain non-data implied that the automaton was operating in the transmit clock domain, hence the reason for the earlier supposition.

4: Latency

An optimal low-latency synchronous retiming is one where the receive clock samples the transmit data at the very beginning of the valid region. The latency for such a retiming is defined to be zero.

A plesiochronous retiming circuit samples either the Q or R waveforms uniformly along their valid regions. When the width of the valid region approaches the entire cell-time, the average latency sampling the Q waveform is 0.5 cell-times. Similarly, the average latency

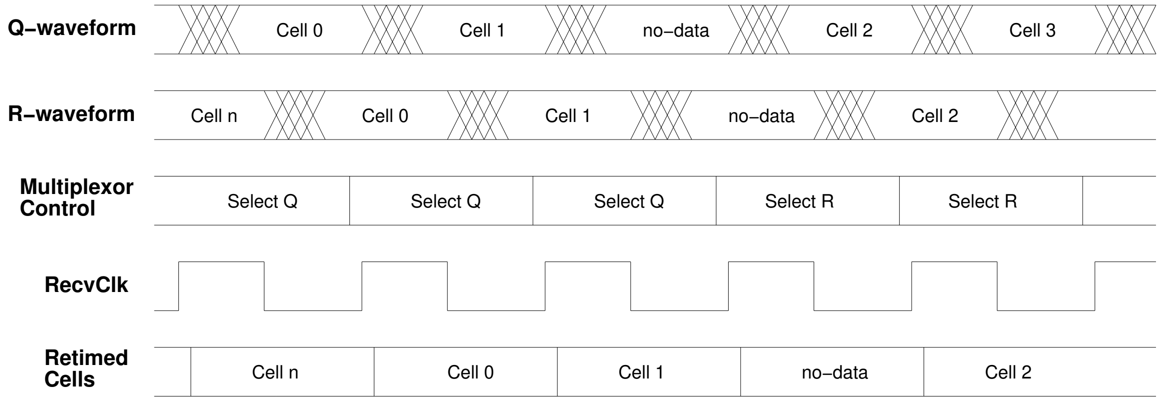

Figure 10: RxClk slow, $Q \rightarrow R$

sampling the R waveform is 1.0 cell-times. Thus, the average latency is 0.75 cell-times.

However, when the width of the valid region drops to about half a cell-time, the latency for sampling the Q waveform drops to 0.25 cell-times and the latency for sampling the R waveform drops to 0.75 cell-times. The average drops to .5 cell-times.

A *biased* waveform selection finite state machine would try to use the Q waveform whenever possible. Under these conditions, the average latency is 0.5 cell-times.

5: Circuit Pragmatics

This section describes the circuits used in the Reliable Router. There are many possible good implementations of plesiochronous communications, so this should be used as guidance only. The circuits involved are used to regain timing margin between devices, to construct the Q and R waveforms, to detect the exclusion region, and to control waveform selection.

Please note that the unit of transfer commonly used by communication engineers is a *cell*. In the parallel processing community, the term *flow control control digit* or *flit* is used. Since plesiochronous retiming does not require flow control, descriptions of router operations are presented in terms of cells rather than flits.

Between reliable routers, a transmit clock is sent along with a cell. As mentioned in the retiming section, several timing events in the transmit clock domain are required per cell. Needed are:

- Q goes into valid
- Q goes into exclusion
- R goes valid
- R goes into exclusion
- Change between Q and R now.

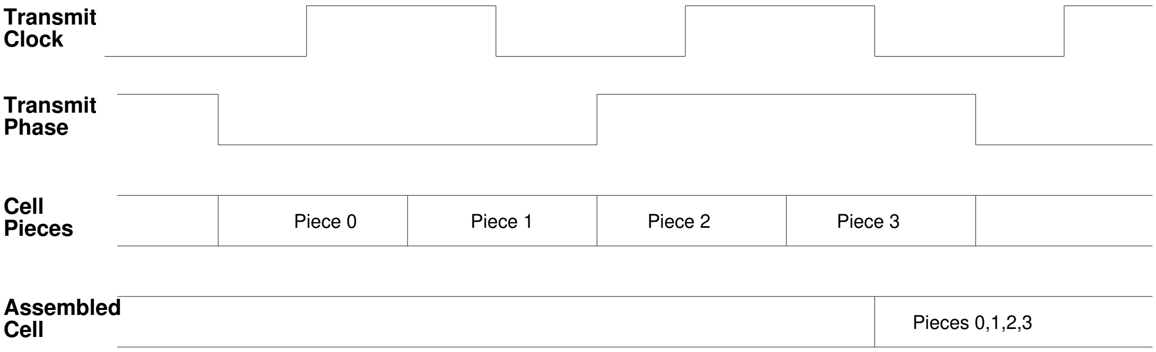

Figure 11: Cell reassembly

For many reasons unrelated to plesiochronous communications, in the router a cell is broken into four smaller pieces, each transmitted on four subsequent transmit clock edges. Both positive and negative edges are used, so a cell time is two transmit clock periods (the router uses a 100Mhz transmit clock). A transmit phase signal is used to distinguish piece number zero from piece number two, and piece number one from piece number three. The edge of the transmit clock indicates whether the piece is even or odd. These pieces are reassembled by the receiver into a complete cell. A sketch of the waveforms in shown in figure 11.

When the cell is finally reassembled, the actual exclusion region for the Q waveform is relatively small and straddles a transmit clock edge. The other three transmit clock edges can be used to construct the R waveform and control the $Q \leftrightarrow R$ crossing point.

The discussion of the exclusion region detection was a little simplified. One does not want to wait until the receive clock is sampling in the exclusion region before deciding to change! Instead, a keep-out region which encompasses the exclusion region is constructed using transmit clock edges. The transmit clock edges thus represent:

- R goes into keep-out, Q goes into valid
- R changes
- Q goes into keep-out, R goes into valid, change mux control here.
- Q changes

Figure 12 shows the waveforms and their relative timing.

To determine if a change between Q and R waveforms is required, the keep-out regions are first sampled using the receive clock. The output of these sampling flip-flops are then fed back into the transmit clock domain using a synchronizer for use by the automaton.

The entire automaton has two states: select-Q and select-R, and is shown in figure 13.

Note that the synchronizer delay does not delay the movement of data. "Stale" in-keepout information is perfectly okay, as long as the relative clock drift is not too fast. Since the delay is out of the critical path, fast metastability resolution is not required of the synchronizer. In the router, 100ns are alloted to synchronizer resolution.

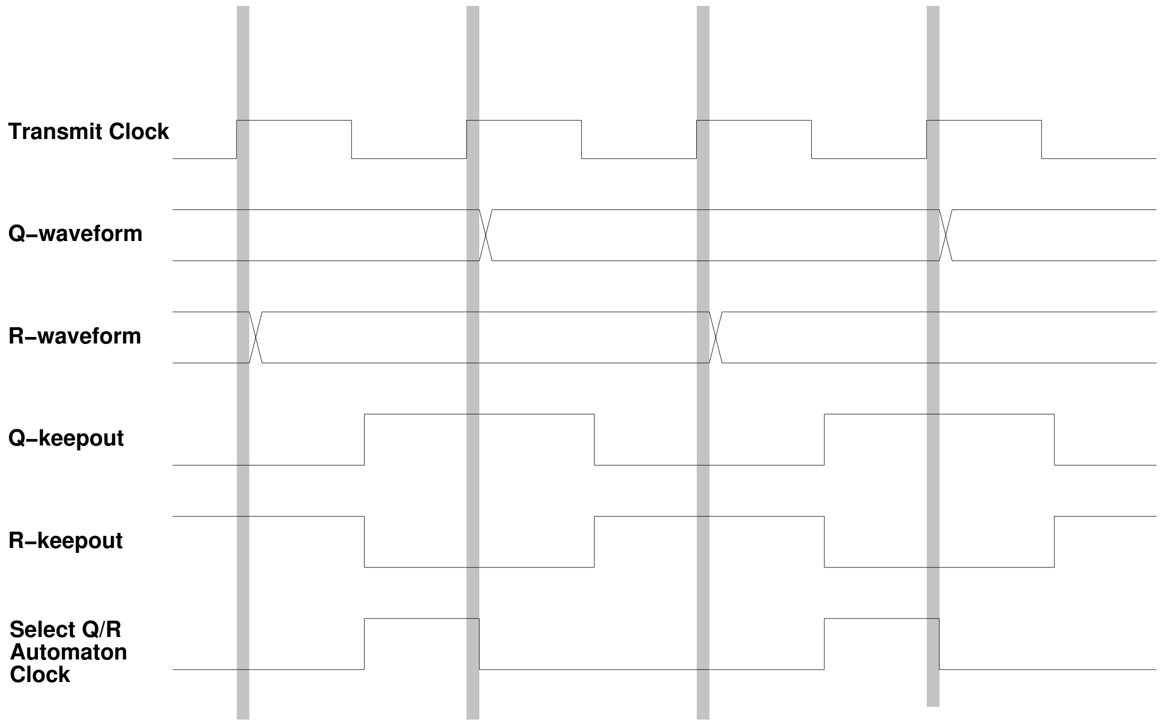

Figure 12: Waveforms derived from transmit clock

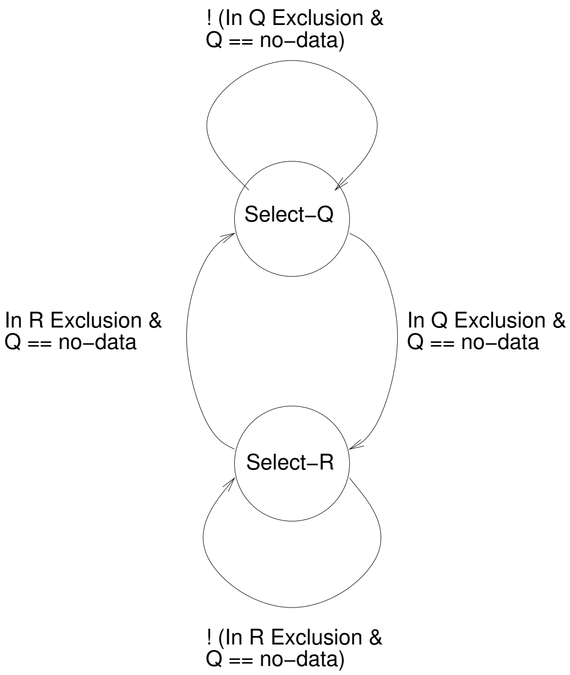

Figure 13: Select $Q \leftrightarrow R$ Finite State Diagram

One of the nice features of this implementation is that no circuit tricks were required. Slowing down the clock will improve timing margins, just in case one missed a critical path.

5.1: Non-Data Transmission Rate

In the previous section, a distinction between the exclusion region and the keep-out region was made. A non-data cell must be received somewhere in the time it takes the receive clock edge to drift from the edge of the keep-out region to the edge of the exclusion region. This time is the maximum amount of time allowed between transmission of non-data cells.

To begin the calculation, the time from keepout region edge to exclusion region edge, $t_{k\rightarrow e}$, is expressed as a phase angle:

$$\Psi = 2\pi t_{k\rightarrow e} f_r \tag{3}$$

From equation 2, the change in phase angle with respect to a change in time is:

$$\Delta\Theta(\Delta t) = 2\pi f_t - f_r \Delta t$$

For worst case f_t and f_r:

$$\Delta\Theta(\Delta t) = 4\pi \Delta f \Delta t \tag{4}$$

Setting 3 and 4 equal

$$4\pi \Delta f \Delta t = 2\pi t_{k\rightarrow e} f_r$$

$$\Delta t = t_{k\rightarrow e} \frac{f_r}{2\Delta f}$$

$$\Delta t = t_{k\rightarrow e} \frac{(f_0 - \Delta f)}{2\Delta f}$$

Assuming $\Delta f \ll f_0$:

$$\Delta t \approx t_{k\rightarrow e} \frac{f_0}{2\Delta f} \tag{5}$$

Converting into transmit cell times:

$$\#\ \text{cell times} \approx t_{k\rightarrow e} \frac{f_0}{2\Delta f} f_t$$

$$\#\ \text{cell times} \approx t_{k\rightarrow e} \frac{{f_0}^2}{2\Delta f} \tag{6}$$

For example, suppose that the keep-out region exceeds the exclusion region by 2ns, the base frequency is 50Mhz (i.e. 50 million cells per second), and that the accuracy is ± 5khz (100 ppm).

$$\frac{2e\text{-}9 \times 50e6 \times 50e6}{2 \times 5e3} = 500 \text{ cell times}$$

In this example, 0.2% of the available bandwidth must be given over to non-data.

6: Integral Subrate Extensions

The method may be extended to allow retiming between clock domains where the frequency of one domain is an integral multiple of the other domain. That is, either $f_t \approx i \times f_r$ or $f_r \approx i \times f_t$ must hold.

When $f_r \approx i \times f_t$, the circuit changes are fairly minor. The receiver must sample both the keep-out windows and the incoming Q or R cell once every i periods. For clock periods when the incoming cell is not sampled, the receiver creates a non-data cell.

If $f_t \approx i \times f_r$, the transmitter must drastically reduce its sending of data to one of i cells. Note that it still must occasionally insert non-data cells to meet the plesiochronous requirements.

This technique is used in the Reliable Router to allow slower-speed processors to interface easily to a higher speed router.

7: Summary

A new technique for plesiochonous data retiming has been described. This technique offers latencies on the order of a fraction of a cell-time as well as modest implementation requirements. These acheivements were gained by moving the synchronizer out of the data path and carefully choosing the time to make a phase adjustment.

In addition, the technique allows true unidirectional retiming. The transmitter can send information to a receiver without any flow control information sent back to the transmitter from the receiver.

Extensions include the ability to handle integral subrates. This should prove to be useful in the design of systems where multiple clock rates are present.

Acknowledgments

The authors wish to express their sincere thanks to the past and present members of the Reliable Router team: Jeffrey Bowers, Dan Hartman, Kin Hong Kan, Ivan Oei, and David Harris.

References

[1] W.J. Dally, L.R. Dennison, D. Harris, K. Kan, T. Xanthopoulos, "Architecture and Implementation of the Reliable Router," In *Hot Interconnects II: 1994 Symposium Record.*

[2] D.G. Messerschmitt, "Synchronization in Digital System Design", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 8, October 1990.

[3] R.D. Rettberg and L.A. Glasser, U.S. Patent 4,700,347, October 13, 1987.

[4] W.K. Stewart and S. Ward, "A Solution to a Special Case of the Synchronization Problem," *IEEE Transactions on Computers*, vol. 37, no. 1, January 1988.